Joni Tauriainen

SMS-RAJAPINNAN TOTEUTUS IOS-YMPÄRISTÖSSÄ

SMS-RAJAPINNAN TOTEUTUS IOS-YMPÄRISTÖSSÄ

Joni Tauriainen
Opinnäytetyö
Kevät 2015
Ohjelmistokehityksen koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehitys

Tekijä(t): Joni Tauriainen
Opinnäytetyön nimi: SMS-rajapinnan toteutus iOS-ympäristössä
Työn ohjaaja(t): Johannes Nikula, Ouman Oy; Pertti Heikkilä, Oulun ammattikorkeakoulu
Työn valmistumislukukausi ja -vuosi: kevät 2015 Sivumäärä: 45 + 3 liitettä

Tässä opinnäytetyössä tutkittiin mahdollisuutta toteuttaa tekstiviestejä itsenäisesti lähettävä sekä vastaanottava sovellus iOS-käyttöjärjestelmälle. Työ toteutettiin Ouman Oy:lle ja sovelluksen pohjana toimi Android-ympäristöön aikaisemmin tehty Fonel-60-sovellus, jolla hallitaan Fonel-60-ohjaus- ja valvontajärjestelmää.

Projektissa perehdyttiin iOS-käyttöliittymän suunnitteluun ja toteutukseen ja tutustuminen iPhoneen aloitettiin perusasioista. Sovellus sekä alustava käyttöliittymä toteutettiin Objective-C-ohjelmointikielellä ja Xcode-kehitystyökaluilla. Sovellus käyttää tekstiviestien lähetyksessä sekä vastaanotossa SMSGlobal-palvelua ja HTTP-kutsuja. Tekstiviestien lukua varten tehtiin myös tietokantapalvelin, josta vastaanotetut viestit voidaan hakea.

Projektin lopputuotteena valmistui iPhone-sovellus, jossa on alustava käyttöliittymä iOS-ympäristössä ja josta voidaan kommunikoida Fonel-60:n kanssa.

Asiasanat: iOS, Objective-C, mobiililaitteet, matkapuhelimet, HTTP

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software Development

Author(s): Joni Tauriainen
Title of thesis: SMS interface implementation for iOS systems
Supervisor(s): Johannes Nikula, Ouman; Pertti Heikkilä, Oulu University of Applied Sciences
Term and year when the thesis was submitted: Spring 2015
Pages: 45 + 3 appendices

In this thesis project, goal was to find a way to let iPhone send and receive text messages within iOS application. Project was carried out by Ouman Ltd. and basis for application was Ouman's Fonel-60 Android application which is used to control Fonel-60 management and security system.

One part of project was to familiarize with designing and developing iOS mobile interfaces and this was done with very basics of designing. Application and it's preliminary interface was made with Objective-C and Xcode-IDE. Application uses SMSGlobal service and HTTP calls to send and receive text messages. Also a database server was made for storing received messages.

Project's end product was a iPhone application which has rough user interface and which can send text messages independently.

Keywords: iOS, Objective-C, mobile-platform, mobile device, HTTP

SISÄLLYS

SANASTO

API – ohjelmointirajapinta, jonka kautta eri ohjelmat voivat keskustella keskenään

HTTP – on selainten ja www-palvelinten käyttämä tiedonsiirtoprotokolla

iOS – mobiilikäyttöjärjestelmä, joka on käytössä Applen mobiililaitteissa

JSON – (JavaScript Object Notation) Yksinkertainen tekstimuotoinen tiedonsiirtomuoto

MySQL – Relaatiotietokantaohjelmisto

Objective-C - Applen käyttämä ohjelmointikieli

Relaatiotietokanta - tietovarasto, jossa tiedoilla on yhteys toisiinsa

String-olio - objekti, joka muodostuu peräkkäisistä merkeistä

Unirest - kevyt monialustainen HTTP-kirjasto

Xcode - kehitystyökalu Applen laitteille

1 JOHDANTO

Opinnäytetyöni aiheena oli tutkia ja selvittää mahdollisuutta toteuttaa Fonel-60-järjestelmän hallintasovellus iPhone-laitteille. Työn suurimpana haasteena oli löytää mahdollisia ratkaisuja lähettää ja vastaanottaa tekstiviestejä iOS-sovelluksesta, sillä iOS-käyttöjärjestelmä ei salli sovelluskehittäjille pääsyä tekstiviestitoimintoihin sovelluksen sisältä. Tärkeimpänä tavoitteena työlle oli löytää ainakin yksi tapa lähettää ja vastaanottaa tekstiviestejä iPhone-sovelluksen sisältä. Opinnäytetyössä päädyttiin tutkimaan erilaisia web-palveluita, jotka pystyisivät lähettämään ohjelmasta syötetyn tekstiviestin ja säilyttämään mahdollisen vastausviestin lukemista varten.

Opinnäytetyö toteutettiin projektiluontoisesti Ouman-konsernille, jossa oli aikaisemmin kehitetty vastaava hallintasovellus Android-laitteille. Tuo sovellus toimi lähtökohtana iOS-versiolle sovelluksesta. Opinnäytetyössä tutustuttiin myös yleisesti iPhone-sovelluksen kehittämiseen Applen tarjoamilla työkaluilla sekä sivuttiin tietokantapalvelimen rakentamista.

Työn alkuperäisenä suunnitelmana oli toteuttaa valmis ja Android-sovellusta vastaava tuote, joka sisälsi samat ominaisuudet ja toiminnot. Kuitenkin jo työn alussa kävi ilmi, ettei iOS-ympäristö salli tekstiviestin lähetystä tai saatujen viestien lukemista sovellusten sisältä samalla tavalla kuin Android-käyttöjärjestelmä, jolloin opinnäytetyön luonnetta päätettiin muuttaa tutkimusluonteiseksi. Lopputuotteena opinnäytetyöltä oli sovellus, joka sisältää Android-sovellusta vastaavat Mittaukset- sekä Asetukset-näkymät ja jossa tekstiviestien lähettäminen ja vastaanotto on toteutettu SMSGlobal-palvelua ja HTTP-kutsuja käyttäen.

2 FONEL-60-JÄRJESTELMÄ

Fonel-60 iPhone -sovelluksen kehitys aloitettiin Ouman Oy:llä FSM Groupin pyynnöstä tutkia mahdollisuutta tehdä iOS-versio mobiilisovelluksesta, koska yrityksen mukaan kysyntää olisi myös iPhone-versiolle Fonel-60-sovelluksesta.

2.1 Ouman Oy

Ouman on vuonna 1988 perustettu kiinteistöjen automaatioon ja energiatehokkuuteen erikoistunut suomalainen konserni. Konserni työllistää yhteensä 250 henkilöä 12 eri toimipaikalla ympäri Suomea ja lisäksi kolmella ulkomaalaisella toimipaikalla. Tehtaat sijaitsevat Kempeleessä ja Viron Kuresaaressa. Konsernin liikevaihto oli 33 miljoonaa euroa vuonna 2014. (1; 2.)

Käytännössä Ouman valmistaa älykästä ja helppokäyttöistä kiinteistöautomaatiota, jotka myydään konsernin omalla tuotemerkillä. Tänä päivänä yritys panostaa myös energiatehokkuuteen sekä älykkäästä lämmönsäädöstä saataviin säästöihin. Konsernin tarjoamat palvelut voidaan karkeasti jaotella kolmeen luokkaan: energiatehokkuuspalvelut, rakennusautomaatio ja OEM/ODM-palvelut, eli lisensoi ja markkinoi muiden tuotteita omalla brändillään. Tärkeimpiä tuotteita ovat Ouflex-tuoteperhe, Ounet-nettivalvomo, Lämmönvahti sekä OEM-tuotteet. (1; 2.)

2.2 Fonel-60-järjestelmä

Fonel-60 on Oumanin suunnittelema ja valmistama älykäs GSM-ohjaus- ja valvontajärjestelmä, jota FSM Group jälleenmyy. Se on tarkoitettu älykkääksi ratkaisuksi sähkölämmitteisen omakotitalon, kesämökin tai pienen liikekiinteistön ohjaus- ja valvontatarpeisiin ja sen avulla voidaan toteuttaa lämmityksen kaukoohjaus ja -valvonta ajasta ja paikasta riippumatta. Lisäksi sillä voidaan toteuttaa muun muassa rakennuksen murto-, palo- ja vesivuotovalvontaa tai ohjata älykkäästi autonlämmitystä. Järjestelmän etäohjaus ja -hallinta tapahtuu matkapuhelimella tekstiviestein. (Kuva 1.) (3, s. 2.)

KUVA 1. Fonel-60 -järjestelmä kotelossaan

Kommunikointi Fonel-60-järjestelmän kanssa tapahtuu tekstiviesteillä avainsanojen avulla ja se tapahtuu periaatteella kysy-muokkaa-lähetä. Laitteeseen voidaan asettaa määrityksiä ja arvoja eri toiminnoille ja tapahtumille tekstiviestien avulla. (Kuva 2.) (3, s. 2.)

14.3 Fonel Fonel-60:ssä on käyttöönotettavissa seuraavat toiminnot

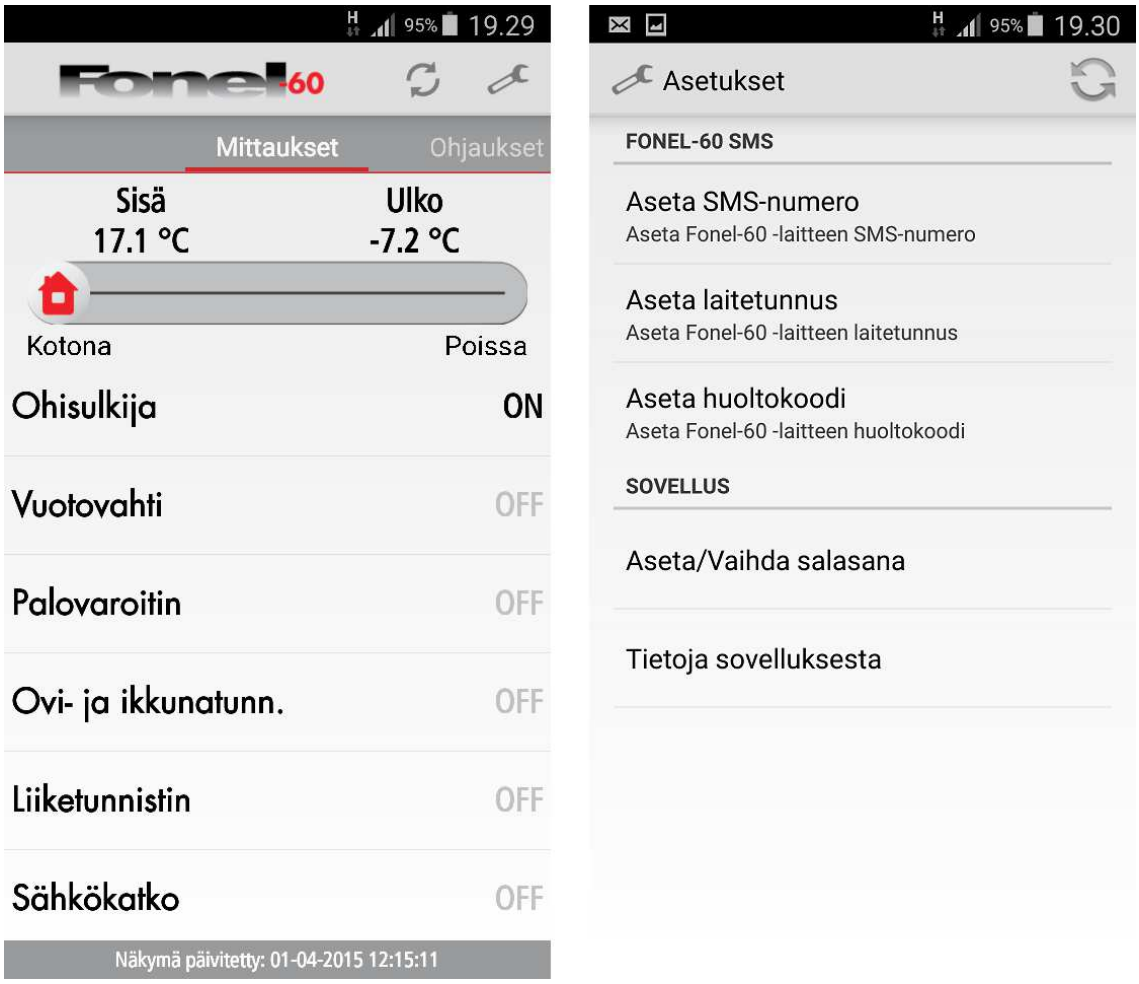
Toiminnon nimi	Lyhyttunnus	Lisätietoa toiminnosta
Termostaattitoiminta	TERMO	Huonelämpötilan ohjaus kotona/poissa-tilaan. Aseta lämpötilat Fonel-60:llä (ks. s. 11)
Jäätymisvaara	JÄÄTV	Jäätymisvaarahälytys huonelämpötilan laskiessa alle raja-arvon (ks. s. 11)
Kuorivalvonta	KUORI	Murtovalvonta ovikytkimien ja lasirikkoilmaisimien avulla (ks. s. 4)
Tilavalvonta	TILAV	Murtovalvonta liikeilmaisimien avulla (ks. s. 4)
Vuotovahti	VUOTO	Vesivuotovalvonta vesivuotoilmaisimia käyttäen (ks. s. 6)
Palohälytys	PALOH	Palohälytys saadaan kosketintietona palovaroittimilta (ks. s. 6)
Lämmönpud.ohjaus	PUDOT	Huonelämpötilan ohjaus kotona/poissa-tilaan. (ks. s. 10)
Autolämmitys	AUTOL	Autopistorasioiden toiminta ulkolämpötilan ja kellon ohjaamana (ks. s. 12)
Yleisohjaus	YLEIS	Ohjaus vuosi/viikkokellolla, ajastimella tai pakko ohjauksella (ON/OFF) (ks. s. 12)
Ulkolämpötila	ULKOL	Ulkolämpötilan mittaus -toiminto. Otetaan käyttöön haluttaessa seurata ulkolämpötilaa, eikä autolämmitystoimintoa ole otettu käyttöön.
Tehonsyöttö		Toiminto voidaan ottaa käyttöön, jos käytössä on akkuvarmennus (ks. s. 8 ja 19)
IV-ohjaus	IVOHJ	IV-ohjaus tapahtuu murtovalvonnan päälle/ pois päältä (ohisulkija) kytkimen ohjaamana. (ks. s. 21)
Ovilukitus	OVILU	Ovilukitus tapahtuu murtovalvonnan päälle/ pois päältä (ohisulkija) kytkimen ohjaamana.

26

KUVA 2. Fonel-60-laitteen käyttöönotettavat toiminnot (3, s. 26)

2.3 Fonel-60 Android -sovellus

Fonel-60 Android -sovellus (kuva 2) on Oumanilla kehitetty applikaatio, jolla kommunikointi Fonel-60-laitteen kanssa esitetään käyttäjäystävällisessä muodossa käyttöliittymässä. Se on ladattavissa veloituksetta Android Marketista. (3, s. 2.)



KUVA 3. Android-sovelluksen Mittaukset- sekä Asetukset-näkymät

Sovelluksen tarkoitus on helpottaa Fonel-60-järjestelmän hallitsemista ja poistaa käyttäjältä tarve tutustua monimutkaiseen avainsanoin kommunikointiin. Se muotoilee käyttäjän antamista komennoista halutun tekstiviestin ja lähettää sen laitteelle. Vastausviestistä se lukee saadun vastauksen ja päivittää saadut tiedot käyttöliittymään. Esimerkkinä kuvassa 3 on saatu talon sisä- ja ulkolämpötila järjestelmältä vastauksena. Ohjelma myös vastaanottaa välittömästi tiedon palo-, kosteus- ja murtohälytyksistä ja huomauttaa käyttäjää varoituksen saadessaan Androidin ilmoituspalkin ilmoituksella. Ohjelman voi salata asettamalla henkilökohtaisen salasanan, jolloin ulkopuoliset eivät pääse valvontajärjestelmän ohjaukseen käsiksi (4). (3, s. 2.)

Sovelluksen käyttöliittymä on rakennettu käyttäen pitkälti Androidin omia komponentteja. Päänäkymä sisältää viisi erilaista alinäkymää, joita vaihdellaan pyyhkäisyin tai navigointipalkista koskettamalla haluttua näkymää. Alinäkymä muodostuu listasta, johon on koottu näkymän sisältämät arvot. Muokattavaa riviä klikkaamalla aukeaa dialogi-ikkuna, josta voidaan valita uusi arvo kyseiselle riville. Jos arvoa on muutettu, ylös aukeaa uusi palkki, josta voi hyväksyä tai peruuttaa muutokset. Muutoksien hyväksyntä käynnistää prosessin, jossa muutetuista arvoista lähetetään viesti Fonel-laitteelle. Normaalisti sivun yläosassa on palkki, josta voi päivittää aktiivisen alinäkymän tai siirtyä Asetukset-näkymään. Alinäkymän päivitys kysyy näkymän tiedot Fonel-60-järjestelmältä ja tuo vastausviestistä saadut arvot ja tiedot alinäkymään. Asetukset-näkymästä voidaan muuttaa esimerkiksi Fonel-60-laitteen numeroa tai huoltokoodia ja lisätä salasanasuojaus mobiiliohjelmaan.

3 KOMMUNIKOINTI JA TYÖN SUUNNITTELU

3.1 Kommunikointi Fonel-60-järjestelmän kanssa

Kommunikointi Fonel-60-järjestelmän kanssa tapahtuu tekstiviestein avainsanojen avulla. Järjestelmälle lähetetään ensin viestinä haluttu avainsana, johon laite vastaa avainsanan mukaisesti. Joillain avainsanoilla voidaan hakea yhtä tai useampaa arvoa järjestelmältä ja joihinkin avainsanoihin voidaan liittää arvoja, joilla muutetaan jotain asetusta laitteesta. (3, s. 20–21.) Laite myös lähettää tarvittaessa oma-aloitteisesti viestejä asetettuihin numeroihin jonkin hälytyksen kytkeytyessä päälle. Esimerkiksi palohälytyksen aktivoituessa hälytysviesti lähetetään välittömästi palolaitokselle sekä kiinteistönhoitajalle, joiden numerot on ennalta syötetty laitteeseen. (3, s. 8.)

Kommunikointi Fonel-60:n kanssa edellyttää laitetunnuksen ja huoltokoodin käyttöä. Laitetunnuksella varmistetaan, etteivät ulkopuoliset pysty käyttämään laitetta luvatta. Huoltokoodin takaa löytyvät kriittisimmät sekä vähemmän käytetyt laitteen asetukset. Nämä kentät voidaan myös muuttaa, kunhan käyttäjä tietää kyseisen laitteen huoltokoodin ja laitetunnuksen. (3, s. 23.)

3.1.1 Esimerkki mittausten lukemisesta

Taulukko 1 on esimerkki yksinkertaisesta mittauskyselystä.

TAULUKKO 1. Esimerkki mittausten lukemisesta laitteelta

Lähetetty viesti	Vastausviesti(t)
FO60 0000 MITTAUKSET	FO60 MITTAUKSET: M1 K-P kytkimen tila= Kotona *Poissa/ M2 Ohisulkija= *ON OFF / M3 Vuotovahti=OFF/ M4 Palovaroitin=OFF/ ...jatkuu... FO60 MITTAUKSET: ...jatkuu... M5 Ulkolämpötila=-7.2C/ M6 Huonelämpötila=17.1C/ M7 Ovi- ja ikkunatunn.=OFF/ M8 Liiketunnistin=OFF/ ...jatkuu... FO60 MITTAUKSET: ...jatkuu... M9 Sähkökatko=OFF

Mittauksia luettaessa lähetettävä viesti muodostuu kolmesta osasta:

[LAITETUNNUS] <[HUOLTOKOODI]> [AVAINSANA]

Esimerkissä FO60 on laitetunnuksen tehdasasetus, kuten myös huoltokoodi 0000. Nämä asetukset ovat myös vaihdettavissa laitteesta, mutta tässä raportissa käytetään näitä tehdasasetuksia, jos yhteydessä ei muuten mainita. Huoltokoodi on tässä toiminnossa vaihtoehtoinen kenttä, koska sitä ei tarvita käyttäjätason toiminnoissa. MITTAUKSET on taulukko 1:n avainsana, jolla kerrotaan Fonel-järjestelmälle, että viestin lähettänyt numero odottaa paluuviestinä kaikkien laitteeseen kytkettyjen mittauskanavien lukuarvoa tai tilaa. (3, s. 18.)

Vastausviestejä voi olla useita Fonel-60-laitteeseen asetettujen toimintojen ja mittausten nimien pituuden mukaan. Vastausviesti alkaa aina laitetunnuksella sekä avainsanalla, joka kertoo tilatun toiminnon. Avainsanan jälkeen mahdollisesti tuleva ”...jatkuu...”-merkkijono kertoo, että viesti on jatkoa johonkin edelliseen viestiin, sama merkkijono viestin lopussa taas indikoi, että viestiin on tulossa jatkoa. Viestissä näkyy mittauskanava, johon kyseinen mittaus on kytketty, ja mittauksen nimi (esimerkiksi M2 Ohisulkija) sekä lukuarvo tai tilatieto. Fonel-60:n mittaukset ovat analogisia, jotka ovat joko NTC-mittauksia tai potentiaalivapaita kosketintietoja (tilatietoja). Kosketintieto kertoo mittauksen sen hetkisen tilan ja on yleensä muotoa ON/OFF (päällä/pois). Joillain kentillä voi olla myös poikkeava kosketintieto; esimerkiksi taulukko 1:ssäkin esiintyvä Kotona-Poissa-kytkin on tilat KOTONA/POISSA. Kytkintiedoissa se tila, jonka edessä tähtimerkki sijaitsee, on parhaillaan aktiivinen tila. (3, s. 18.)

3.1.2 Esimerkki asetusarvojen muuttamisesta

Taulukko 2 on esimerkki yhden asetusarvon muuttamisesta.

TAULUKKO 2. Esimerkki asetusarvon muuttamisesta laitteelta

Lähetetty viesti	Vastausviesti(t)
FO60 0000 ASETUSARVOT: PALOH Palohälytys: HÄL:VIIVE=3s	FO60 0000 Muutettu: ASETUSARVOT: PALOH:HÄL:VIIVE=3

Mittatuloksia haettaessa lähetettävä tekstiviesti muodostuu seuraavasti:

[LAITETUNNUS] [HUOLTOKOODI] [AVAINSANA]: [ASETUSARVO]=[UUSI ARVO]

Asetusarvojen muuttaminen tapahtuu tekstiviestin muokkaustilassa, jolloin muutettava asetusarvo ja sen uusi arvo tulevat avainsanan jälkeen. Tässä muutoksessa on käytettävä huoltokoodia, koska kyse on huoltotason asetusarvosta ja laite ei anna muuttaa arvoa, mikäli huoltokoodia ei toimiteta viestin mukana.

Fonel-60 lähettää vielä tekstiviestinä vahvistukseksi uudet päivitetyt asetukset. (3, s. 20.)

3.1.3 Esimerkki kotona-poissa-kytkimen ohjauksesta

Kotona-poissa-kytkimen tila voidaan saada selville taulukko 3:n viestillä.

TAULUKKO 3. Esimerkki kotona-poissa-kytkimen tilatiedon lukemisesta

Lähetetty viesti	Vastausviesti(t)
FO60 TILATIETO	FO60 TILATIETO: M1 K-P kytkimen tila=Kotona *Poissa/ M2 Ohisulkija=*ON OFF

Jos Fonel-60-järjestelmässä on murtovalvontatoiminto, näkyy vastausviestissä myös ohisulkijan tila. Näiden kytkinten tilaa voidaan muuttaa muokkaustilassa helposti muuttamalla vastausviestinä saadusta viestistä tähtimerkin paikkaa haluttuihin tiloihin ja lähettämällä viesti laitteelle. Esimerkiksi Kotona-poissa-kytkin saadaan KOTONA-tilaan ja Ohisulkija OFF-tilaan lähettämällä seuraava viesti:
FO60 TILATIETO: M1 K-P kytkimen tila=*Kotona Poissa/ M2 Ohisulkija=ON *OFF
Pelkkää kotona-poissa-kytkimen tilaa voidaan helposti hallita lähettämällä laitteelle yksinkertaisesti viesti FO60 KOTONA tai FO60 POISSA. (3, s. 10.)

3.2 Suljetun iOS API:n vaikutukset

Applen tarjoamat tuotteet, mukaan lukien sovelluskehityksen työkalut ja kirjastot, ovat pääsääntöisesti niin sanottuja suljettuja ympäristöjä. Tämä tarkoittaa, ettei Apple anna kehittäjille eikä muillekaan oikeutta muuttaa mitään käyttöjärjestelmien sisältä ja yritys julkaisee kehittäjille vain API:n, joka tarjoaa työkalut sovelluskehitykselle, mutta jonka pääsy eri käyttöjärjestelmän toimintoihin on Applen rajoittama. (5.) Tämä lähestymismalli eroaa huomattavasti Googlen omistamasta Android-järjestelmästä, joka on vastaavasti avoimeen lähdekoodiin perustuva ympäristö, eli kaikki sen ominaisuudet ovat kehittäjien saatavilla sekä muokattavissa. (6.)

Tämä ero on otettava huomioon, kun tehdään sovellusversioita eri mobiilikäyttöjärjestelmien välille. Mikä onnistuu yhdessä käyttöjärjestelmässä, ei välttämättä ole mahdollista toisessa. Lisäksi käyttöjärjestelmillä on omat käytänteet ja mallit eri komponenttien käytöstä; suoraan Androidista kopioitu käyttöliittymä voi tuntua oudolta iOS-ympäristössä. Yleensä käyttöliittymän suunnittelussa tehdään omat käyttöliittymät eri käyttöjärjestelmille käyttämällä kohdelaitteen omia käyttöliittymäkomponentteja. (7.)

Kun Fonel-60-mobiilisovelluksen kääntämisessä Androidilta iPhonelle alettiin suunnitella, esille nousi välittömästi ongelma: Apple ei ole avannut iOS:n tekstiviestien lähetystä taikka vastaanottoa kehittäjilleen. Tekstiviestin lähetys onnistuu niin pitkälle, että ohjelmasta voidaan pyytää iOS:n omaa iMessage-tekstiviestisovellusta avautumaan ja syöttää ohjelmasta valmiiksi viestikenttään haluttu viesti sekä valita vastaanottajan numero, jolloin käyttäjän tehtäväksi jäisi vain lähetyksen hyväksyminen. iOS-ympäristössä ei ole kuitenkaan mitään tapaa päästä käsiksi laitteen vastaanottamiin tekstiviesteihin, jolloin Fonel-60-sovelluksesta ei olisi suoraa tapaa päästä käsiksi laitteen vastausviesteihin. Syynä tähän on luultavasti se, että Apple ei halua tarjota sovelluskehittäjille mahdollisuutta tehdä ohjelmia, jotka voisivat lähettää viestejä sovelluksen taustalla ilman, että käyttäjällä olisi tietoa tästä. (8.)

3.3 Opinnäytetyön suunnittelu

Opinnäytetyön aikana tuotettiin kaksi eri suunnitelmaa: tutkimussuunnitelma sekä projektisuunnitelma. Työn luonne ehti vaihtua näiden kahden dokumentin teon välillä

3.3.1 Tutkimussuunnitelma

Tutkimussuunnitelma löytyy raportin liitteestä 1. Tutkimussuunnitelmaa tehdessä opinnäytetyön alkuperäisenä tavoitteena oli tutustua iOS:n SMS-ominaisuuksiin ja niiden käyttöön ja saada lopputuotteeksi valmis iOS-käyttöliittymä. Tässä vaiheessa ei ollut vielä täysin tiedossa, että iPhonen SMS-rajapinta on kokonaan suljettu kehittäjiltä. Työn tarkoituksena oli siis perehtyä iPhone-käyttöliittymän tekemiseen ja jättää tekstiviestikommunikaatio taka-alalle.

Tässä vaiheessa myös lopullinen ohjelmointikieli oli jätetty avoimeksi, eli vaihtoehtoina olivat Objective-C- sekä Swift-kielet. Projektihallintaan ehdotettiin Scrum-viitekehystä taikka kevyempää Scrum-bania.

Dokumentti toimi asiakirjana josta voitiin tehdä päätös kehityksen käynnistämisestä ja saada selville resurssit, jotka oli hankittava sovelluksen toteuttamiseksi.

3.3.2 Projektisuunnitelma

Projektisuunnitelma valmistui kokonaisuudessaan projektin 1. sprintin aikana ja sitä ei luonteensa ja pituutensa vuoksi oli lisätty liitteeksi tähän raporttiin. Lopullisen suunnitelman tekohetkellä oli kuitenkin jo vahvistunut tieto siitä, että iOS-ympäristössä ei ole mahdollista toteuttaa tekstiviestikommunikaatioita sellaisenaan ja että tekstiviestien lähetykselle sekä luvulle joudutaan keksimään vaihtoehtoinen toteutustapa.

Työn aiheeksi on kirjattu tähän dokumenttiin, että työn tarkoituksena on tutkia ja löytää tapa toteuttaa tekstiviestikommunikaatio Fonel-60-kohdelaitteen ja iPhonen välillä ja samalla suunnitella sekä toteuttaa ainakin käyttöliittymän aloitusnäkymä. Opinnäytetyössä päädyttiin käyttämään Scrum-viitekehystä ja jakamaan aikataulu viiteen osaan; perehtymisvaiheeseen, 1. sprinttiin, 2. sprinttiin, 3. sprinttiin sekä lopetusvaiheeseen. Jokainen sprintti sisältäisi oman tehtävälistansa, johon päivitettäisiin työn etenemistä sekä alitehtäviä, ja jokaisen sprintin aikana pidettäisiin aloitus- sekä lopetuspalaverit. Lisäksi lopetusvaiheeseen oli sovittu loppukatselmointi, jossa esiteltäisiin työn tulosta sekä kulkua. Sprinttisuunnitelmaa päivitettäisiin OAMK:n Redmine-palveluun.

4 KÄYTETYT TEKNIIKAT JA OHJELMISTOT

Sovelluksen kehittämisessä käytettiin Applen tarjoamien työkalujen lisäksi Unirest-kirjastoa, joka on Objective-C-kirjasto HTTP-kutsuille. Lisäksi tekstiviestipalveluna käytettiin SMSGlobal-nimistä palvelua ja käytetty tietokantapalvelin sijaitsi OAMK:n students-palvelimella.

4.1 iPhone-sovelluskehitys

Sovelluskehitys iOS-käyttöjärjestelmälle onnistuu helpoiten Applen omilla Mac-tietokoneilla. Mac-koneen lisäksi kehitys vaatii aktiivisen henkilökohtaisen tai yrityksen omistaman ”iOS Developer Program” -lisenssin, joka maksaa 99 dollaria ja on uusittava vuosittain. (9.) Kehitystä voi tehdä ilman lisenssiä, mutta tällöin ohjelmaa ei ole mahdollista testata fyysisessä iPhone-laitteessa eikä julkaista Applen App Storessa (10). Sovelluskehitys tapahtuu yleensä Xcode-kehitysympäristössä (11).

4.1.1 Objective-C

Objective-C-ohjelmointikielen kehittivät Brad Cox sekä Tom Love 1980-luvun alkupuoliskolla yrityksessään Stepstonessa. Kehittäjät olivat tutustuneet Smalltalk-ohjelmointikieleen ja lainasivat siitä vaikutteita Objective-C:hen, muun muassa oliomallin. Myöhemmin Steve Jobs lisensoi Objective-C:n NeXT-nimiselle yritykselle, jossa kielelle julkaistiin kirjastoja, kääntäjä sekä käyttöliittymäeditori. Yrityksen avoin NeXTStep-käyttöjärjestelmä perustui näihin Objective-C kirjastoihin. Tästä vapaasta käyttöliittymästä kirjastoineen muodostui OpenStep-standardi. Vuonna 1996 Apple osti NeXTtin, päätyi yritys käyttämään tätä standardia uuden Mac OS X -käyttöjärjestelmän perustana. (12.) Applen kehittäjille julkaisemaa Mac OS X API-kirjastoa kutsutaan Cocoaksi ja tämän iOS-vastine on nimeltään Cocoa Touch (13).

Objective-C on pääsääntöinen ohjelmointikieli, kun kirjoitetaan sovelluksia Applen OS X - ja iOS-ympäristöön. Se on C-ohjelmointikielen laajennus, joka mahdollistaa oliomallin sekä dynaamisen ajosuorituksen. Objective-C perii syntak-

sin, primitiivityypit sekä ohjauslohkorakenteen perus-C-kieleltä, mutta lisää siihen syntaksin luokille sekä luokkametodeille. Se myös lisää tuen luokkaperinälle sekä literaalityypille. (14.)

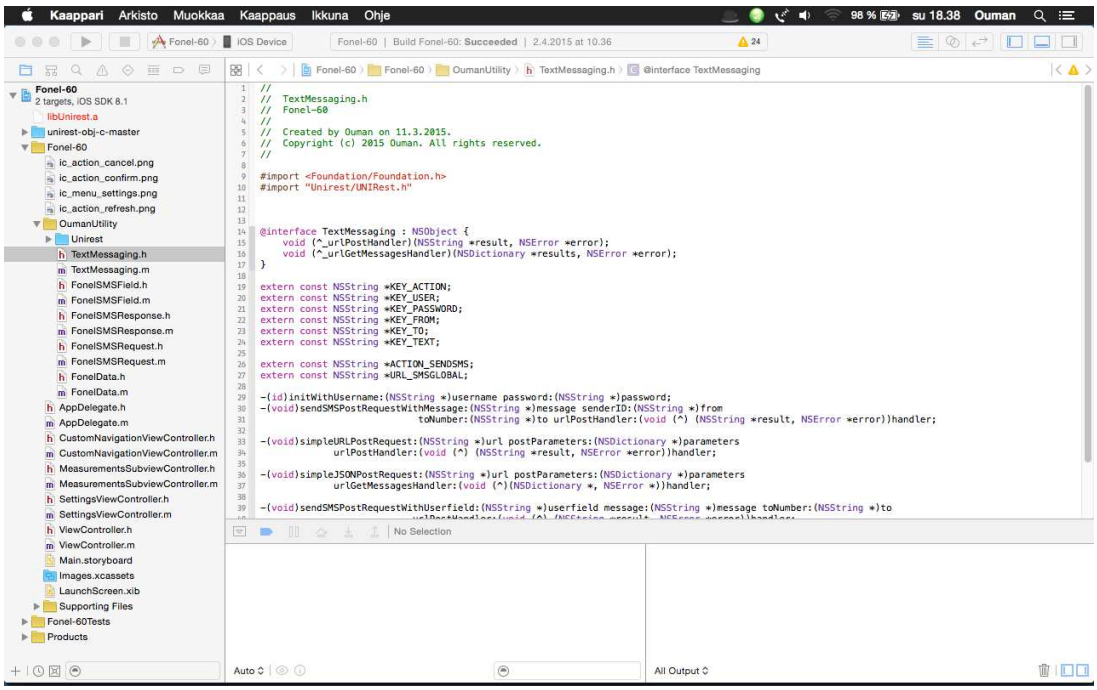
Muista oliopohjaisista kielistä, kuten C++:sta, Objective-C eroaa siten, että luokkametodeita ei suoranaisesti kutsuta, vaan luokan objektille lähetetään viesti suorittaa haluttu metodi (taulukko 4) (12).

TAULUKKO 4. Esimerkki C++- ja Objective-C-kielten metodien käytön eroista

C++	Objective-C
`objekti->metodi(argumentti);`	`[objekti metodi:argumentti];`

4.1.2 Xcode-kehitysympäristö

Xcode-työkalusarjaa käytetään iOS- ja Mac OS X -sovellusten kehittämiseen (kuva 4). Se sisältää työkalut lähdekoodin muokkaamiselle, virheenpaikannukselle, käyttöliittymän suunnittelulle, suorituskyvyn analysoinnille, sovelluksen simuloimiselle sekä julkaisulle. Ympäristön lataa helpoiten Macin App Storen kautta. Asennuksen mukana tulee kaikki tarvittavat työkalut ja kirjastot sovellusten kehittämiseen, mutta laajempi kehitystyö vaatii lisenssin hankkimisen. (15.)

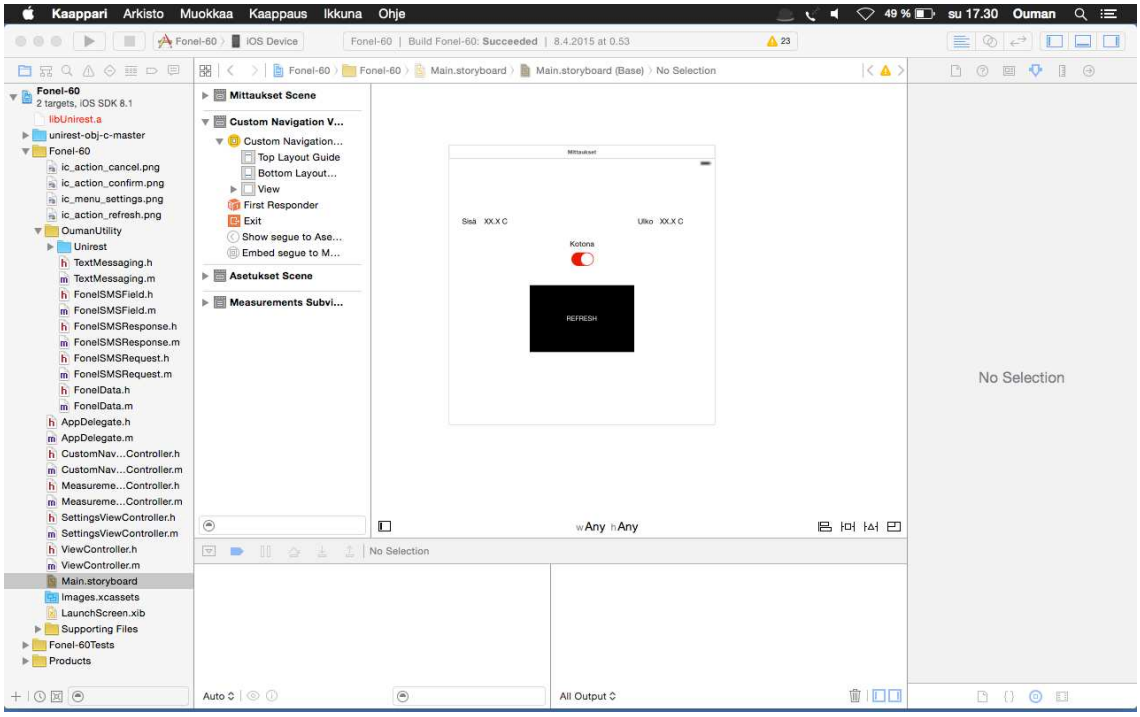


KUVA 4. Kuvakaappaus Xcode-kehitysympäristöstä

4.1.3 Interface-Builder

Xcode sisältää myös käyttöliittymäeditorin (kuva 5), jolla voidaan suunnitella ja toteuttaa mobiilikäyttöliittymiä ilman koodin kirjoittamista. Se toimii vedä ja pudota -periaatteella, eli käytettävät komponentit raahataan alarivin palkista näkymään. Apple käyttää mallia, jossa käyttöliittymät rakennetaan erillään koodista ja eri komponentit voidaan yhdistää muuttujiin, jotka yhdistetään ajon aikana koodiin ja joita voidaan näin ollen hyödyntää itse toteutuksen puolella. (16.)

Käyttöliittymän rakennetaan useista näkymistä, joiden välillä käyttäjä navigoi. Editori luokin eri näkymien välille yhteyden, joista muodostuu niin sanottu ohjelman käsikirjoitus. (16.)



KUVA 5. Kuvakaappaus Xcoden Interface-Builder -työkalusta

4.2 Unirest for Objective-C

Unirest on kevyt HTTP-kirjasto ja se on käännetty useille eri ohjelmointikielille. Objective-C-käännös kirjastosta sisältää useita hyödyllisiä toimintoja, kuten HTTP Get- ja Post-kutsujen käyttö ohjelmasta. Kirjasto tukee niin kutsuttuja asynkronisia kutsuja, jolloin eri web-kutsuja voidaan suorittaa omasta säikeestään, ilman että kutsuilla vaikutetaan ohjelman käyttöliittymän toimintaan. Kirjaston mukana tulee myös automaattinen JSON-parseri, jonka avulla voidaan helposti käsitellä nettiin lähetettävää sekä netistä luettavaa tietoa. (17.)

4.3 SMSGobal-palvelu

SMSGlobal on kansainvälinen web-palvelu, joka tarjoaa erilaisia mobiiliviestinnän palveluita. Se on perustettu vuonna 2007 ja sillä on 33 työntekijää neljällä toimipisteellä eri puolilla maailmaa. Yrityksen palvelut toimivat parhaillaan 213 eri maassa ja asiakkaisiin kuuluu muun muassa maiden hallituksia, lentokenttiä ja kansainvälisiä brändejä. (18.) Yksi yrityksen tarjoamista palveluista on HTTP-API, jolla voidaan yksinkertaisesti toteuttaa tekstiviestin lähetys ja vastaanotto web-palvelun kautta. (19.)

Palvelusta on mahdollista ostaa kuukausimaksullinen sopimus, jolla käyttöönsä saa puhelinnumeron ja yhden viestin hinta on halvempi. Toinen vaihtoehto on ladata rahaa tililleen, josta lähetetyt viestit maksetaan. Kuukausimaksullisia paketteja on tarjolla erilaatuisia sekä -hintaisia. Esimerkiksi Advanced-paketti maksaa 36,45 € kuukaudessa ja siihen kuuluu muutama lisäpalvelu. Advanced-paketissa sadan tekstiviestin hinta Finnet 2G -operaattorin kautta olisi 2,85 €. Kuukausimaksuttomassa Basic-paketissa vastaava hinta on 3,31 €. (20.)

4.4 Tietokantapalvelin

Sähköinen tietokantapalvelin muodostuu yleensä itse tietokannasta sekä dynaamisesta web-hallintasysteemistä ja -scripteistä. Yleisimmin käytetään MySQL-tietokantaa sekä PHP-pohjaista järjestelmää, koska ne saadaan pyörimään lähes kaikissa ympäristöissä ja PHP:hen on olemassa lukuisia valmiita kirjastoja MySQL-tietokannan hallinnalle. (21.)

4.4.1 MySQL

MySQL on maailman toiseksi käytetyin relaatiotietokanta ja perustuu avoimeen lähdekoodiin. Ensimmäinen versio MySQL:stä kehitettiin Ruotsissa vuonna 1995 MySQL AB -nimisen yrityksen alla. Vuonna 2008 Sun Microsystems osti oikeudet MySQL:ään ja vuonna 2009 kyseinen yritys päätyi Oracle Corporationin omistukseen. (22.)

MySQL on relaatiotietokanta, jolloin taulujen välille luodaan yhteyksiä niin sanottujen avaimien avulla. Yleisimmin käytetään luotua ID:tä eli numeroyhdistelmää, jolla tietokannan rivit erottuvat toisistaan. (23.) Avoimen lähdekoodin alla MySQL:stä on julkaistu monia versioita eri käyttöjärjestelmälle, joten se toimii lähes missä tahansa palvelinympäristössä. Lisäksi sitä pidetään nopeana, luotettavana ja suhteellisen helppokäyttöisenä. (21.) MySQL:stä onkin muodostunut tavallisin tietokantasysteemi, kun halutaan säilöä paljon tietoa. Muun muassa Facebook, Twitter ja Wikipedia käyttävät sitä tietokannoissaan. (22.)

4.4.2 PHP

PHP on ohjelmointikieli, jota käytetään erityisesti web-pohjaisissa sivustoissa ja palvelimissa. Se on käytetyin dynaamisten web-palveluiden tuottamiseen tarkoitettu kieli, tammikuussa 2013 PHP oli asennettuna yli 240 miljoonalle web-sivustolle. (24.)

Kielen ensimmäisen version julkaisi vuonna 1995 grönlantilainen Rasmus Lerdorf ja vuosien saatossa kieli on saanut lukuisia päivityksiä ja parannuksia. PHP-skriptiä voidaan sekoittaa perus HTML-koodin sekaan ja käyttää yhdessä muiden skripitkielien kanssa web-sivustojen toimintojen toteutuksessa. Kun selain hakee web-sivua palvelimelta, palvelin suorittaa PHP-koodit itsenäisesti ja lähettää vasta sitten valmiin sivun käyttäjälle. Näin käyttäjä ei pysty urkkimaan PHP-koodia selaimesta käsin, koska koodi suoritetaan palvelimella. Toisaalta PHP:llä voidaan avata vakavia tietoturva-aukkoja jos sitä käytettäessä ei olla varovaisia, koska käyttäjä voi kuitenkin selaimen kautta vaikuttaa tiettyihin para-

metreihin jotka suoritetaan palvelimella. Myös pitkien koodirivien ja suuren tietomäärän käsittely palvelimella voi viedä aikaa, jolloin sivuston lataus selaimessa on hidasta. (24.)

PHP:tä käytetään myös hyvin yleisesti palvelimien tietokantojen käsittelyssä. Palvelimelle ladattava PHP-paketti pitääkin yleensä sisällään luokkakirjaston ainakin MySQL-tietokantojen käsittelylle. Lisäksi sillä voidaan toteuttaa selainpohjaisia tietokantapalvelimia, joita hallitaan PHP-skriptien avulla. PHP-skripti voi ottaa parametrikseen HTTP Post-kutsuna annettuja arvoja, jotka voidaan käsitellä koodissa. Näillä parametreilla voidaan hallita esimerkiksi mitä tietoja halutaan hakea tietokannasta. Palvelin voi näin palauttaa käyttäjälle tietokannasta saadut tiedot esimerkiksi JSON-muodossa, joka on yleisesti käytetty avoimen standardin tiedostomuoto tiedonvälitykselle. (21; 25.)

5 TOTEUTUS

5.1 Aloitus

Projektin aloituspäivinä projektin lähtökohdat olivat vielä tutkimussuunnitelman varassa ja lähtökohtana oli saada valmis käyttöliittymä tehtyä projektin aikana. Ensimmäisenä työnä oli koota toimiva Fonel-60-ympäristö, jota voitaisiin käyttää testauslaitteistona projektia tehdessä (kuva 6). Kokoamisen lisäksi tehtävälistalla oli laitteeseen tutustuminen. Hyvää opastusta tähän tarjosi Fonel-60-käyttöopaskirjanen, jossa kerrottiin kaikki laitteen tärkeimmät ominaisuudet, tekstiviestien lähetys ja tulkinta, laitteen toiminta sekä esiteltiin Android-sovellus. Tässä vaiheessa oli hyvä käydä kaikki ominaisuudet läpi Android-sovelluksesta ja suunnitella niiden kääntämistä iPhone-ympäristöön.

KUVA 6. Fonel-60-testilaitteisto

Ennen varsinaisen suunnittelu- ja tutkimustyöhön ryhtymistä aloitettiin iOS-alustaan tutustuminen esimerkkisovellusten sekä Objective-C-oppaiden avulla. Samalla saatiin kokemusta Xcode-kehitysympäristön ominaisuuksista ja käytöstä.

Objective-C osoittautui hiukan odotettua haasteellisemmaksi kieleksi omaksua, sillä sen syntaksi ja muutamat muut yksityiskohdat eroavat huomattavasti muista suosituista ohjelmointikielistä. Myös käyttöliittymän toteutus eroaa Xcode-kehitystyökalussa muista vastaavista siten, että eri komponentit yhdistetään vetämällä syötettyihin muuttujiin, joita voidaan sitten käyttää ohjelmakoodissa.

Harjoitusten kautta käytiin läpi tarpeellisia tekniikoita tulevan käyttöliittymän suunnittelua varten ja erinäisten kokeiluiden pohjalta alkoi muotoutua näkemys siitä, millaisia ratkaisuilla sovelluksen käyttöliittymää kannattaisi lähteä toteuttamaan.

5.2 SMS-rajapintaan tutustuminen ja sen ongelma

Projektin alkupuoliskolla oli myös oleellista tutustua iOS API:n tarjoamiin rajapintoihin ja ominaisuuksiin ja näistä eritoten SMS-palveluiden käyttöön liittyvää rajapintaa. Projektin alkutietojen pohjalta oli selvää, että iOS-ympäristö tarjoaisi ongelman SMS-rajapintansa kanssa ja että tämän rajapinnan käyttöä joudutaan selvittämää aluksi tarkoin. Applen tarjoamiin dokumentteihin tutustuttaessa kävi erittäin nopeasti selväksi, että tekstiviestien lähettäminen ja lukeminen sovelluksesta käsin olisi sellaisenaan lähes mahdotonta.

Kyseisen seikan selviämisen jälkeen asiasta keskusteltiin palaverissa. Tapaamisessa todettiin, että projektin luonnetta muutetaan ja hylätään suunnitelma, että projektista valmistuisi valmis tuote. Sen sijaan projektissa päätettiin perehtyä löytämään ratkaisu tähän ongelmaan, eli löytämään keino lähettää sekä lukea tekstiviestejä iPhone-puhelimesta.

SMS-ominaisuuksien toteutus tapahtuisi puhelimen web-palveluita käyttäen, mutta reittejä ominaisuuden toteuttamiseksi oli useita. Suunniteltuja vaihtoehtoja oli kolme. Näistä yksi oli rakentaa Oumanille oma palvelin, jolla oli SIM-omi-

naisuudet eli kyky lähettää ja vastaanottaa viestejä. Tähän palvelimeen voitaisiin olla yhteydessä puhelimitse ja lähettää sekä lukea tekstiviestejä sen kautta HTTP-kutsuja käyttäen.

Toisena vaihtoehtona esille nousi mahdollisuus hyödyntää sähköpostipalvelua ja käyttää sitä tekstiviestien lähettämiseen ja vastaanottamiseen. Ainakin Google tarjosi mahdollisuuden lähettää Gmail-tunnuksilla tekstiviestejä tiettyjen operaattoreiden numeroihin ja vastaus viestiin tuli niin ikään Gmail-tilille. Ohjelmaan voitaisiin siis lisätä luokka, jonka kautta voitaisiin kirjautua Gmail-tilille, lähettää sähköpostiviesti puhelinnumeroon ja lukea vastauksena saatu viesti.

Viimeinen ja todennäköisesti luotettavin vaihtoehto ongelmaan oli ostaa tekstiviestipalvelu joltain yritykseltä, joka tarjoaisi web-pohjaisen API:n tekstiviestien lähetykselle sekä vastaanotolle. Tämä vaihtoehto oli myös Oumanin näkökulmasta paras, koska siihen ei tarvittaisi kovin monimutkaisia palvelimia jotka jäisivät yrityksen ylläpidettäväksi tulevaisuudessa. Lisäksi sähköpostin käytössä ratkaisuna ilmeni ongelma, että suomalaiset operaattorit eivät tarjoa julkisina tietoina web-yhdyskäytäviä joiden kautta sähköpostiviesti voitaisiin lähettää tekstiviestinä.

Lisäksi valitun ratkaisun pitäisi tuli olla sellainen, ettei Fonel-60-järjestelmään tarvinnut tehdä muutoksia. Laitteessa on tehdaspaketissaan valmiiksi ohjelmoituna perus käyttäytyminen, mutta siihen on mahdollista myös lisätä omaa toiminnallisuutta. Toiveena oli kuitenkin, ettei iOS-versio vaatisi mitään muutoksia valmiiseen laitteeseen.

5.3 HTTP API

Seuraavana askeleena projektissa oli ryhtyä tutkimaan mahdollisia palveluita, jotka täyttäisivät seuraavat kriteerit:

- Palvelulla on HTTP API, jolla mahdollista kaksisuuntainen viestintä, eli tekstiviestin lähetys sekä vastausviestin lukeminen.
- Käyttö ei saisi olla kovin kallista, eli joko sopiva kuukausimaksu tai yksittäisten tekstiviestien hinta pieni.
- Palvelun käyttö testaukseen olisi halpaa, mielellään täysin ilmaista.

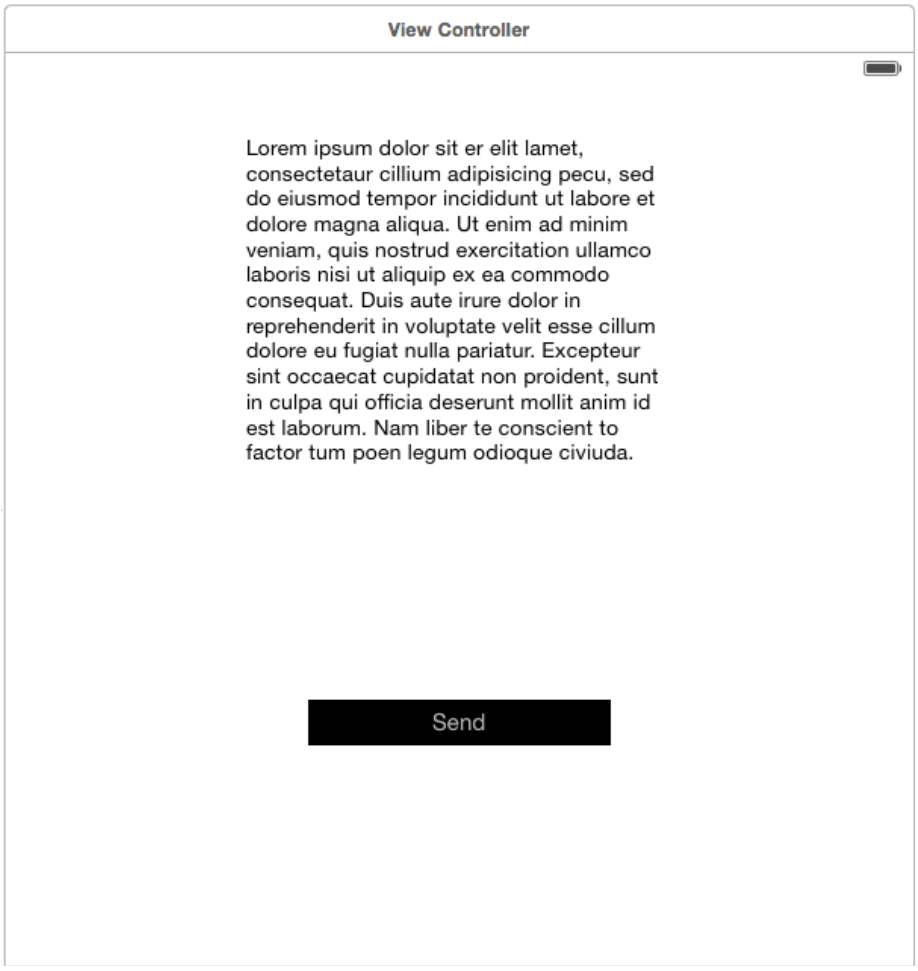
Nettihaun perusteellä löydettiin varsin nopeasti yksi palvelu nimeltä SMSGlobal, jonka tarjoamat palvelut vastasivat hyvin hakemiamme ominaisuuksia. Toisena vaihtoehtona oli Twilio-niminen palveluntarjoaja, joka kuitenkin osoittautui hankalaksi ja hintavaksi käytettäväksi pienimuotoisessa projektissa.

SMSGlobal tarjosi yksinkertaisen HTTP API:n SMS-viestien lähetykselle, sille voitiin ilmoittaa web-palvelin jonne vastaanotetut viestit halutaan säilöä ja lisäksi palvelussa yksittäisen tekstiviestin hinta on kotimaisten operaattoreiden hintojen tasolla. Palvelun käyttö tosin tarkoittaisi ongelmia mahdolliselle loppukäyttäjälle ja tätä pohditaan myöhemmin raportissa.

Valitun SMSGlobal-palvelun sivustolta löytyy kattava opas palvelun eri ominaisuuksiin. Projektissa käytetty HTTP API on kuitenkin suhteellinen yksinkertainen käyttää ja sivustojen tarjoamien dokumenttien avulla pystyttiin alkamaan rakentamaan luokkia joilla voitiin testata tekstiviestin lähetystä iPhone-sovelluksesta.

5.4 Tekstiviestin lähetys

Xcode-projektissa käytettävää SMS-luokkaa alettiin kirjoittaa, kun oli saatu rajattua käytettävä API SMSGlobal-palveluun ja tutkittua hiukan kyseisen API:n toimintaa. SMS-luokka sisältäisi kaikki oleellisemmat metodit tekstiviestien lähetykselle sekä lukemiselle ja metodit käyttäisivät Unirest-kirjastoa ja toimisivat asynkronisesti, jolloin käyttöliittymän toiminnot eivät pysähtyisi web-kutsujen ajaksi. Luokka käyttäisi käyttäjän syöttämiä SMSGlobal-käyttäjätunnuksia, puhelimen puhelinnumeroa sekä Fonel-laitteen puhelinnumeroa, laitetunnusta ja huoltokoodia, joita voidaan muokata valmiista Asetukset-näkymästä. Aluksi nämä arvot olivat kuitenkin kovakoodattuja ja testaamiseen käytettiin väliaikaista testinäkymää. (Kuva 7.)



KUVA 7. Testinäkymä, jolla voitiin testata tekstiviestin lähetystä

Aluksi tärkeintä olikin testata vain palvelun toimivuus ja todeta, että tekstiviesti voidaan onnistuneesti lähettää ohjelman sisältä toiselle laitteelle. Tekstiviestin kulku palveluiden välillä kuvataan liitteessä 2 olevassa kuvassa. Kuvasta nähdään, että iPhone-sovelluksesta lähetetään tekstiviestin parametrit SMSGlobalille HTTP-kutsulla, joka tämän jälkeen hoitaa kyseisen viestin lähetyksen Fonel-60-järjestelmälle.

Tässä projektissa hyödynnetään myös erästä SMSGlobalin ominaisuutta, jossa jokainen sen kautta lähetetty viesti saa lähettäjänumerokseen maakohtaiselta numerolistalta umpimähkään valitun puhelinnumeron. Tätä tapaa käytettäessä palvelu osaa yhdistää valitun numeron käyttäjätunnukseen. Kun vastausviesti saapuu Fonel-laitteelta, pystyy SMSGlobal-palvelu yhdistämään viestin vastaanottajan käyttäjätunnukseen ja lähettämään sen tunnuksiin liitetylle HTML-palvelimelle.

TextMessaging-luokka tulisi sisältämään ominaisuudet tekstiviestien lähetykselle. Luokan metodit siis ottavat parametrikseen tarvittavat tiedot kuten tekstiviestin viestin, vastaanottajan numeron sekä käsittelijämetodin (handlerin), joka on rajapinta kutsuvan luokan kanssa. Tässä rajapinnassa voidaan määrittää miten toimitaan kun viesti on lähetetty sekä tutkia mikäli viestin lähetys epäonnistuu ja toimia sen mukaan. (Esimerkki 1.)

```
-(void)sendSMSPostRequestWithMessage:(NSString *)message senderID:(NSString *)from toNumber:(NSString *)to
                      urlPostHandler:(void (^)(NSString *, NSError *))handler {
    // Tutkitaan onko tarvittavat kentät määritettyinä
    if (apiUsername == nil || apiPassword == nil) {
        NSLog(@"ApiUsername or ApiPassword nil, object not properly initialized");
        return;
    }

    NSDictionary *params = @{   // HTTP-kutsussa lähetettävät parametrit
                                KEY_ACTION: ACTION_SENDSMS,    // Haluttu toiminto
                                KEY_USER: apiUsername,         // SMSGlobalin käyttäjänimi
                                KEY_PASSWORD: apiPassword,     // SMSGlobalin salasana
                                KEY_TO: to,                    // Viestin vastaanottaja
                                KEY_TEXT: message              // Viesti
                                };
    // Kutsutaan metodia joka hoitaa itse viestin lähetyksen asynkronisesti
    [self simpleURLPostRequest:URL_SMSGLOBAL postParameters:params urlPostHandler:handler];
}

-(void)simpleURLPostRequest:(NSString *)url postParameters:(NSDictionary *)parameters
              urlPostHandler:(void (^)(NSString *, NSError *))handler {

    _urlPostHandler = [handler copy];

    // Luodaan headers-taulukko, johon syötetään HTTP-postissa käytettävät kentät sekä
    // metodin parametreina saadut parametrit
    NSDictionary *headers = @{@"accept": @"application/json"};
    [[UNIRest post:^(UNISimpleRequest *request) {
        [request setUrl:@"http://www.smsglobal.com/http-api.php"];
        [request setHeaders:headers];
        [request setParameters:parameters];
    }] asStringAsync:^(UNIHTTPStringResponse *stringResponse, NSError *error) {
        NSLog(@"simpleUrlPostRequest: URL Post done");

        // Kutsutaan postHandler-rajapintaa jos sellainen on määritettynä
        if (_urlPostHandler != nil) {
            _urlPostHandler(stringResponse.body, error);
            _urlPostHandler = nil;
        }
    }];
}
```

ESIMERKKI 1. TextMessaging-luokan tekstiviestin lähetyksessä käytetyt metodit

Esimerkki 2 on katkelma koodista, jota käytettiin TextMessaging-luokan testauksessa. Se luo olion, josta kutsutaan viestin lähetyksen metodia, ja metodille annetaan parametrina lähetettävän tekstiviestin kentät.

```
-(void)refreshView {
    // Mikäli vastauksen odottaminen tai muokkaus on päällä ei tehdä mitään
    if (waitingResponse || modifying) return;
    waitingResponse = true; // Odotetaan vastausta

    // Luodaan TextMessaging -olio
    TextMessaging *txt = [[TextMessaging alloc] initWithUsername:foneldata->smsGlobalUsername
        password:foneldata->smsGlobalPassword];
    // Kutsutaan metodia, joka hoitaa tekstiviestin lähetyksen
    [txt sendSMSPostRequestWithUserfield:@"TESTI" message:@"FO60 MITTAUKSET"
                                 toNumber:foneldata->fonelPhoneNumber
                                 urlPostHandler:^(NSString *response, NSError *error)
    {
        // Luodaan handleri metodille, jota kutsutaan TextMessaging-luokassa
        // kunhan tekstiviestin lähetys on saatu tehtyä
        // Tämä handleri hoitaa näkymän päivityksen kunhan tekstiviesti on lähetetty
        dispatch_async(dispatch_get_main_queue(), ^{
            // Käynnistetään päivitys-säie jos se ei ole jo luotuna
            if (updateThread == nil) {
                [self startUpdate];
            }
        });
    }];
}
```

ESIMERKKI 2. Tekstiviestin lähetys testinäkymästä

5.5 Tietokantapalvelin

SMSGlobal-palvelu mahdollistaa kaksisuuntaisen viestinnän käytön, mutta viestin vastaanottamiseen tulee palvelun käyttäjän pystyttää oma web-palvelin vastaanottamaan ja säilömään tekstiviestejä. Projektia varten päätettiin rakentaa oma tietokantapalvelin, joka käyttäisi tietokantana MySQL-relaatiotietokantaa ja datan käsittelyn hoitamisessa PHP-skriptejä palvelinpuolella. Palvelimella sms_received.php -skripti lisää saadut tiedot tietokantaan.

Yhtenä ominaisuutena SMSGlobalissa on mahdollisuus liittää käyttäjätunnuksien asetusten kautta web-sivun osoite palvelulle muistiin. SMSGlobal ottaa yhteyden tähän sivuun kun se vastaanottaa paluuviestin, joka on osoitettu tälle käyttäjätunnukselle.

Liite 2:ssa näkyy paluusuuntana SMS-vastausviestin kulku. Se siis lähtee normaalina tekstiviestinä Fonel-60-laitteelta ja menee SMSGlobal-serviceen, jossa palvelu osaa yhdistää tekstiviestin vastaanottajanumeron tiettyyn käyttäjätunnukseen. Tämän jälkeen palvelin lukee tähän käyttäjätunnukseen liitetyn www-sivun osoitteen, jossa sijaitsee tietokantapalvelin. Palvelu syöttää HTTP Post -

kutsulla tekstiviestin tiedot palvelimelle, joka luo näistä tiedoista rivin tietokantaan.

Tietokanta ottaa parametreina vastaan ja säilyttää seuraavat tiedot saaduista viesteistä:

- ”to” eli mihin numeroon viesti lähetettiin (satunnainen numero SMSGlobalin numerolistalta)
- ”from” eli viestin vastaanottajan puhelinnumero
- ”msg” eli viestin sisältö
- ”userfield” valinnainen kenttä, jota voidaan tekstiviestiä lähettäessä määrittää ja kenttä kulkeutuu myös paluuviestissä takaisin palvelimelle
- ”date” eli viestin vastaanottopäivämäärä ja kellonaika
- ”readed” eli onko tämä rivi käyty jo lukemassa tietokannasta

5.6 Tekstiviestin lukeminen palvelimelta

Kun saatu vastausviesti Fonel-60-laitteelta on tallennettu tietokantaan, pitäisi se saada luettua ja viestin tiedot siirrettyä käyttäjän nähtäväksi käyttöliittymään. Palvelimelle kirjoitettiin tähän tarkoitukseen get_smses.php -tiedosto, joka vastaanottaa POST-kutsulla tarvittavat parametrit ja palauttaa tietokannasta haetut kentät JSON-muodossa takaisin laitteelle.

Liitteen 1 XML-kaaviossa tätä lukuyhteyttä kuvaa nuoli iPhone-laitteesta MySQL-tietokantapalvelimelle. Yhteys on kaksisuuntainen, joten tämän takia viivan molemmissa päissä on nuoli.

Lukua varten TextMessaging-luokkaan lisättiin uudet metodit, joilla saatiin muodostettua yhteys palvelimeen ja lähetettyä halutut parametrit. Palvelin palauttaisi löydetyt kentät takaisin ja tekstiviestin sisältö voitaisiin muodostaa näistä kentistä. Tietokanta piti lukua siitä, onko tietty rivi jo käyty lukemassa ja ohjelmassa tahdottiin saada vain ne viestit luettua, joita ei ole vielä käyty lukemassa. (Esimerkki 3.)

```objc
-(void)readSMSPostRequestWithNumber:(NSString *)from urlGetMessagesHandler:(void (^)(NSDictionary *, NSError *))handler {

    // Tutkitaan onko tarvittavat kentät määritettyinä
    if (apiUsername == nil || apiPassword == nil) {
        NSLog(@"ApiUsername or ApiPassword nil, object not properly initialized");
        return;
    }

    NSDictionary *params = @{   // HTTP-kutsussa lähetettävät parametrit
                               KEY_FROM: from,    // Numero, josta viesti on saatu
                               KEY_ACTION: [NSString stringWithFormat: @"%ld",
                                   (long)ACTION_GETUNREADED]   // Haluttu toiminto formatoidaan string-muotoon
                              };
    // Kutsutaan metodia joka hoitaa itse viestin lukemisen asynkronisesti
    [self simpleJSONPostRequest:URL_GETSMSES postParameters:params urlGetMessagesHandler:handler];
}

-(void)simpleJSONPostRequest:(NSString *)url postParameters:(NSDictionary *)parameters
       urlGetMessagesHandler:(void (^)(NSDictionary *, NSError *))handler {

    // Luodaan headers-taulukko, johon syötetään HTTP-postissa käytettävät kentät sekä
    // metodin parametreina saadut parametrit
    NSDictionary *headers = @{@"accept": @"application/json"};
    _urlGetMessagesHandler = [handler copy];

    // Tehdään asynkroninen POST-kutsu, joka saa vastaukseksi JSON-taulukon
    [[UNIRest post:^(UNISimpleRequest *request) {
        [request setUrl:url];
        [request setHeaders:headers];
        [request setParameters:parameters];
    }] asJsonAsync:^(UNIHTTPJsonResponse* response, NSError *error) {
        NSDictionary *result = response.body.JSONObject;
        if ([result objectForKey:@"success"]) {
            if (_urlGetMessagesHandler != nil) {
                NSDictionary *messages = [result objectForKey:@"messages"];

                // Jos viestejä löydettiin ja handleri on olemassa, kutsutaan
                // handleria hoitamaan halutut tehtävät
                if (messages != nil)
                    _urlGetMessagesHandler(messages, error);
                _urlGetMessagesHandler = nil;
            }
        }
        else {
//            NSLog(@"No messages found");
        }
    }];
}
```

ESIMERKKI 3. TextMessaging-luokassa käytettävät tekstiviestin lukumetodit

Esimerkki 4 on testinäkymästä, jossa kutsutaan readSMSPostRequestWith-Number-metodia ja sille annetaan parametrina käsittelijä (handler), joka hoitaa tarvittavat tehtävät jos lukemattomia viestejä on löydetty tietokannasta tälle puhelinnumerolle.

```
- (void)updater {
    TextMessaging *txt = [[TextMessaging alloc]
        initWithUsername:VALUE_USER password:VALUE_PASSWORD];
    [txt readSMSPostRequestWithNumber:PHNUMBER_FONEL
        urlGetMessagesHandler:^(NSDictionary *results, NSError *error) {
        // Käydään läpi result-taulukon rivit ja tulostetaan niiden arvot konsoliin
        for (NSDictionary *dic in results) {
            for (id obj in dic) {
                NSLog(@"key: %@     obj: %@", obj, [dic objectForKey:obj]);
            }

            // Tulostetaan viestin sisältö näytölle ilmoituksena
            NSString *message = [dic objectForKey:@"message"];
            NSString *from = [dic objectForKey:@"from"];

            NSString *action = [NSString stringWithFormat:@"Response from %@", from];
            [self showNotificationWithDelay:1 alertAction:action alertBody:message];
        }
    }];
}
```

ESIMERKKI 4. readSMSPostRequestWithNumber-metodin käyttö näkymästä

Koska viesti kulkee eri web-palveluiden läpi, jotka käyttävät stringeissään eri koodausta, on tietokantaan tallennetuista kentistä hävinnyt ä- ja ö-kirjaimet. Tämä on otettu huomioon kun on alettu suunnittelemaan ja toteuttamaan projektin seuraavaa osaa.

5.7 Tekstiviestin parsiminen ja formatointi

Parsimisella tarkoitetaan tässä yhteydessä sitä, kun Fonel-60-laitteelta tekstiviestinä saadut tiedot luetaan ja muutetaan ohjelmalliseen muotoon. Formatointi taas on tekstiviestin muotoilua, joka lähetetään Fonel-60-järjestelmälle.

Tekstiviestin parserointi rakentuu kahdesta eri oliosta: FonelSMSField-luokasta sekä FonelSMSResponse-luokan metodeista. FonelSMSField-luokka pitää sisällään yhden avainsana-arvoparin eli yhden osan Fonel-60-laitteen vastausviestistä ”/”-merkkiin asti. Sille annetaan stringinä yksi avainsana-arvopari ja kun se käsketään tulkata tämä pari, se lukee stringistä kentän tyypin, mittauskanavan sekä kanavan arvon.

FonelSMSResponse taas on luokka, joka pitää sisällään kaikki yhdesä tekstiviestistä saadut tiedot. Se siis sisältää taulukon, jossa on kaikki tekstiviestin avainsana-arvoparit FonelSMSField-muodossa. Sille annetaan stringinä yhden

tekstiviestin sisältö, jonka luokka tulkkaa kenttä kerrallaan, eli avainsana-arvo-pari kerrallaan. Se siis pilkkoo tekstiviestin osiinsa ja tekee jokaisesta osasta FonelSMSField-olion, jonka se syöttää taulukkoon. Se myös tutkii löytyykö viestin lopusta ”...jatkuu...”-rivi, joka siis kertoo, tuleeko tähän viestiin vielä jatkoa. Mikäli tämä rivi löytyy lopusta, voidaan se tieto saada boolean-arvona luokasta luettua.

Viimeinen SMS-kommunikoinnissa käytettävä komponentti on FonelSMSRequest-luokka, jota käytetään kun tahdotaan muuttaa jonkin Fonel-60-laitteen kentän tilaa. Luokassa on yksi julkinen metodi, joka ottaa vastaan muokattavan kentän numeron, kentän uuden tilan sekä käsittelijän. Käsittelijä hoitaa tarvittavat muutokset vastausviestin saapumisen jälkeen. Kentän numeron sekä arvon perusteella metodi parsii tarvittavan stringin ja lähettää tuon stringin TextMessaging-luokan olion kautta Fonel-60-järjestelmälle. Tällä metodilla voidaan muuttaa vain yhden kentän arvoa kerrallaan.

5.8 Käyttöliittymän näkymien suunnittelu ja toteutus

Kun iOS-sovelluksen käyttöliittymää alettiin suunnitella, oli pohjana Android-laitteelle tehty sovellus. Tarkoituksena oli kuitenkin tehdä näkymistä sellaiset versiot, jotka soveltuisivat paremmin iOS-ympäristöön ja käyttäisivät Applen mobiilikäyttöjärjestelmässä yleisimpiä käytettyjä komponentteja. Koska projektin pääpaino ei ollut enää käyttöliittymässä, keskityttiin suunnittelussa ja toteutuksessa vain Mittaukset- sekä Asetukset-näkymiin, joiden Android-version löytyvät aikaisemmin esitetystä kuvasta 2.

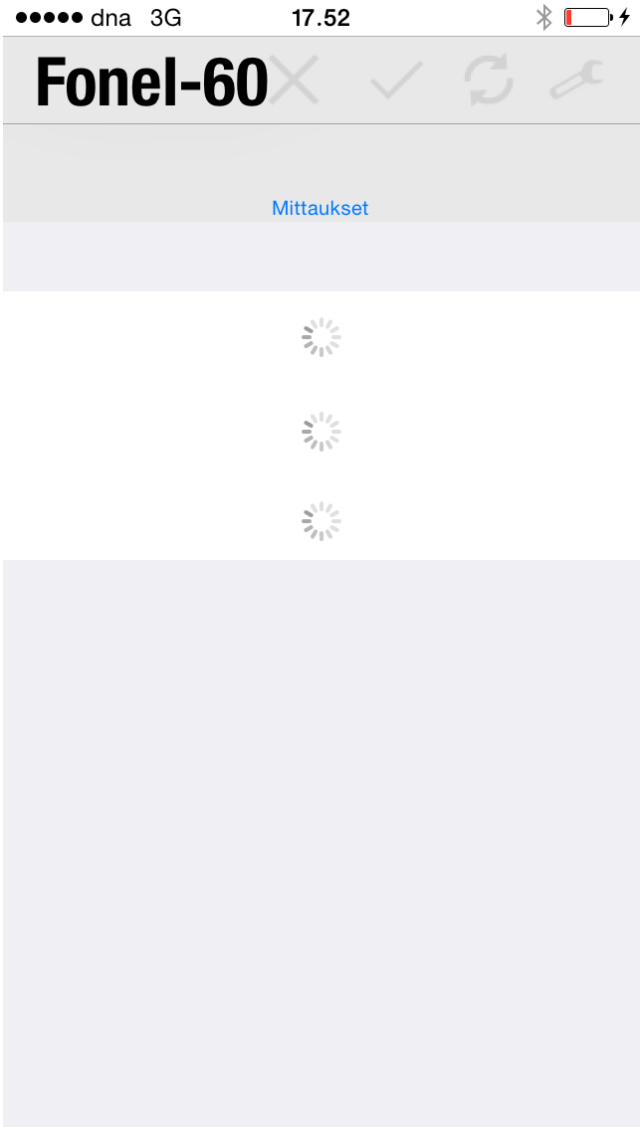
Mittaukset-näkymä pitää iOS-versiossa sisällään lämpötilarivin, tilanapit Kotona/Poissa-tilalle ja Ohisulkija-tilalle. Lämpötilariville tulisi Fonel-60-laitteelta saatujen mittatulosten perusteella sisä- ja ulkolämpötilat. Kotona/Poissa- sekä Ohisulkija-tila päivittyisivät myös mittatulosten perusteella. Näiden kenttien sekä koko näkymän päivitys tapahtuu ylhäällä olevan työkalupalkin kolmannesta painikkeesta. (Kuva 8.)

KUVA 8. iPhone-sovelluksen Mittaukset-näkymä

Tilanapin painaminen käynnistää muokkaustilan, jossa työkalupalkin kaksi ensimmäistä painiketta aktivoituvat ja kaksi viimeistä poistuvat käytöstä. Tilanappien painaminen muuttaa kyseisen mittauksen tilaa. Työkalupalkin ensimmäisen painikkeen painaminen peruuttaa muokkauksen ja tilanapit palautuvat entiseen tilaansa. Toisen painikkeen painaminen hyväksyy tehdyt muutokset, jolloin sovellus siirtyy lähetystilaan. (Kuva 9.)

KUVA 9. Mittaukset-näkymä muokkaustilassa

Lähetystilan kytkeytyessä päälle sovellus lähettää muutettujen tilojen tiedot tekstiviestillä Fonel-60-laitteelle ja jää odottamaan paluuviestinä saatavaa vahvistusviestiä. Tässä tilassa kaikki komponentit kytkeytyvät pois päältä ja käyttöliittymästä ei voida muokata arvoja tai siirtyä muihin näkymiin kun vastausviestiä odotetaan. Koodissa sovellus pyörii päivityssäikeessä, joka käy viiden sekunnin välein hakemassa tulleita viestejä palvelimelta. Sovellus myös pysyy tässä tilassa niin kauan, että viimeinen viesti saadaan palvelimelta vastaanotettua tai niin pitkään, että asetettu maksimiaika täytyy, jolloin lähetystilasta palataan normaaliin toimintaan. Lähetystilan maksimiaika on vakiona 20 sekuntia. (Kuva 10.)



KUVA 10. Mittaukset-näkymä lähetystilassa

Asetukset-näkymään pääsee painamalla Mittaukset-näkymästä työkalurivin viimeistä painiketta. Tähän näkymään voidaan syöttää tarvittavat numerot ja tunnukset jotta sovellus toimisi yhteydessä Fonel-laitteen kanssa. Asetukset tallennetaan FonelData-luokkaan, jossa on staattinen instanssi, jota voidaan käyttää ja hakea tallennetut tiedot mistä tahansa muusta luokasta käsin. Back-painikkeen painaminen työkaluriviltä palaa Mittaukset-näkymään. (Kuva 11.)

KUVA 11. Asetukset-näkymä

5.9 Tietoturva

Projektin loppupuolella oli myös hyvä aika tarkastella ja tutkia sovelluksen tietoturvaa, joka on myös hyvin oleellinen asia sen jatkon kannalta. Lopputuotteena valmistuneessa sovelluksessa ei tietoturvaan ollut kiinnitetty niinkään huomiota ja sellaisenaan käytettynä sen HTTP-ratkaisu voisi olla altis väärinkäytölle ja hakkeroinnille. Koska SMSGlobalille sekä tietokantapalvelimelle lähteviä yhteyksiä ei ole suojattu eikä kulkevaa tietoa salattu millään lailla, voisi asiaan perehtynyt henkilö saada pääsyn tietokantaan ja sen tietoihin palvelimen kautta.

Ratkaisuna tähän olisi salata ja suojata käytetyt yhteydet ja mahdollisesti muuttaa käytettyä arkkitehtuuria sen verran, että myös lähtevä viesti menisi ensin tietokantapalvelimen kautta. Palvelin hoitaisi sen välityksen joko itse tai lähettäisi käytetylle SMS-palvelulle. Liitteessä 3 oleva XML-kaavio on tehty tämän muutoksen pohjalta, ja se kuvaa järjestelmää, jossa kaikki kutsut tehtäisiin oman palvelimen kautta. Lisäksi HTTP-kutsujen sijaan palvelimen ja puhelimen välillä käytettäisiin SSH-yhteyttä, joka on salattuun tietoliikenteeseen tarkoitettu protokolla. SSH:ta käytettäessä puhelimelle luotaisiin oma salattu avain palvelimelle, jota käyttämällä saisi ainoastaan luotua datayhteyden puhelimen ja palvelimen välille. Sovellus voisi sitten käyttää tätä yhteyttä tekstiviestin lähetykseen sekä vastausviestien lukemiseen ja kaikki tieto kulkisi laitteiden välillä salatusti.

6 YHTEENVETO

Opinnäytetyön tavoitteena oli tutkia ja löytää tapa lähettää sekä lukea tekstiviestejä iOS-ympäristössä sovelluksen sisältä sekä tutustua iPhone-käyttöliittymän suunnitteluun ja toteutukseen. Näitä tietoja käytettiin alustavan Fonel-60-sovelluksen tekemiseen, jolla pystytään kommunikoimaan Fonel-60-järjestelmän kanssa. Ouman-konserni tulee käyttämään tämän työn tuloksia päättäessään Fonel-60 iOS -sovelluksen jatkosta.

Työ saavutti sille asetetut tavoitteensa ja siinä todistettiin, että iOS-sovelluksesta on mahdollista lähettää tekstiviestejä sekä lukea vastausviestejä ulkoisen palvelun avulla. Lisäksi opinnäytetyö toteutettiin alustava käyttöliittymä ja arvioitiin tietoturvaa sovelluksen jatkon kannalta.

Jatkon kannalta sovelluksella on kuitenkin haasteensa. Käytetty SMSGlobal-palvelu ei välttämättä ole paras ratkaisu käytettäväksi, koska se jättäisi paljon tekemistä asiakkaan päähän, kuten tunnuksien luomisen ja ylläpidon. Sen käyttö tulisi myös aiheuttamaan lisäkustannuksia jollekin osapuolelle. Lisäksi koska sovellus käyttää kommunikoinnissa jatkuvasti internet-yhteyttä, on tietoturvaa hyvä arvioida ja tutkia paremmin. Vaikka ohjelman niin sanottu hakkerointi ja hyödyntäminen on epätodennäköistä, on sekin uhkakuva kuitenkin otettava huomioon.

Aikaisempi kokemukseni Applen tuotteista oli hyvin vähäinen ja iOS-alustalle ohjelmointi oli minulle täysin tuntematonta. Opinnäytetyön aihe oli siksi minulle hyvin kiinnostava, koska se tarjoaisi mahdollisuuden tutustua uuteen käyttöjärjestelmään, kehitysympäristöön sekä koko joukkoon uusia kehitystekniikoita. Iso osa työtä meni erilaisia vaihtoehtoja tutkiessa ja kokeillessa sekä SMS-palveluihin tutustuessa. Lopputuloksena työlle saatiin kuitenkin aikaiseksi sovellus, jolla voidaan todeta tekstiviestikommunikointi mahdolliseksi iPhone-laitteissa.

LÄHTEET

1. Ouman infojuliste FI1. Saatavissa: http://ouman.fi/wp-content/uploads/2014/10/Ouman_infojuliste_FI1.pdf. Hakupäivä 17.5.2015.

2. Ouman on kiinteistöjen automaatioon ja energiatehokkuuteen erikoistunut suomalainen konserni, joka kasvaa Itämeren alueella. 2015. Ouman. Saatavissa: http://ouman.fi/yritys/. Hakupäivä 17.5.2015.

3. Fonel-60 asennus ja käyttöohje, versio 1.1. FSM Oy. Saatavissa: https://www.dropbox.com/sh/wu0pwoznnbplj7g/AAA7bIG-Majz6b4rbdds5niO4a/Fonel-60%20asennus-%20ja%20k%C3%A4ytt%C3%B6ohje.pdf?dl=0. Hakupäivä 17.5.2015.

4. Fonel-60 - Android-sovellukset Google Playssä. 2015. Google. Saatavissa: https://play.google.com/store/apps/details?id=fi.ouman.eh60&hl=fi. Hakupäivä 17.5.2015.

5. Vidal, Chris. Comparison between open source and closed source software. NSI. Saatavissa: http://info.nsiserv.com/network-support-computer-services-CT/bid/29956/Comparison-between-open-source-and-closed-source-software. Hakupäivä 17.5.2015.

6. Comparison of open source and closed source. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Comparison_of_open_source_and_closed_source. Hakupäivä 17.5.2015.

7. Pungartnik, Heidi. 2014. Differences between iOS 7 and Android 4. Despreneur. Saatavilla: http://despreneur.com/differences-between-ios-7-and-android-4/. Hakupäivä 17.5.2015.

8. Davis, Adam. 2008. Kommentti viestiin ” How to programmatically send SMS on the iPhone?”. Stack Overflow. Saatavissa: http://stackoverflow.com/questions/10848/how-to-programmatically-send-sms-on-the-iphone. Hakupäivä 18.5.2015.

9. iOS Developer Program. 2015. Apple Inc. Saatavissa: https://developer.apple.com/programs/ios/. Hakupäivä 18.5.2015.

10. iOS. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/IOS. Hakupäivä 18.5.2015.

11. iOS Dev Center. 2015. Apple Inc. Saatavissa: https://developer.apple.com/support/ios/ios-dev-center.PHP. Hakupäivä 18.5.2015.

12. Objective-C. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Objective-C. Hakupäivä 18.5.2015.

13. Cocoa (API). 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Cocoa_(API). Hakupäivä 18.5.2015.

14. About Objective-C. 2014. Apple Inc. Saatavissa: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html. Hakupäivä 18.5.2015.

15. Xcode IDE. 2015. Apple Inc. Saatavissa: https://developer.apple.com/xcode. Hakupäivä 18.5.2015.

16. Interface Builder Build-In. 2015. Apple Inc. Saatavissa: https://developer.apple.com/xcode/interface-builder. Hakupäivä 18.5.2015.

17. Unirest for Objective-C. Unirest. Saatavissa: http://unirest.io/objective-c.html. Hakupäivä 18.5.2015.

18. About Us - This is what we're all about. SMSGlobal. Saatavissa: http://www.smsglobal.com/about/. Hakupäivä 18.5.2015.

19. HTTP API - Our documentation, simple, easy. SMSGlobal. Saatavissa: http://www.smsglobal.com/http-api/. Hakupäivä 18.5.2015.

20. Down to Earth - Pricing just for you. SMSGlobal. Saatavissa: http://www.smsglobal.com/pricing/. Hakupäivä 18.5.2015.

21. PHP MySQL Database. W3schools.com. Saatavissa: http://www.w3schools.com/php/php_mysql_intro.asp. Hakupäivä 18.5.2015.

22. MySQL. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/MySQL. Hakupäivä 18.5.2015.

23. Relational database management system. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Relational_database_management_system. Hakupäivä 18.5.2015.

24. PHP. 2015. Wikipedia. Saatavissa: http://fi.wikipedia.org/wiki/PHP. Katsottu 18.5.2015.

25. JSON. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/JSON. Katsottu 18.5.2015.

LIITTEET

Liite 1. Lähtötietomuistio

Liite 2. XML-laitteistokaavio

Liite 3. XML-tietoturva laitteistokaavio

Oulun Ammattikorkeakoulu
Tekniikan yksikkö, Kotkantie
Joni Tauriainen

20.02.2015

TUTKIMUSSUUNNITELMA

Lähtökohdat

Opinnäytetyöni aiheena on Applen iOS -käyttöliittymän suunnittelu ja kehittäminen Fonel-60 tuotteeseen Android-käyttöliittymän pohjalta. Työn tilaajana toimii Ouman Oy:n ja kehitys tapahtuu yrityksen tarjoamilla laitteilla ja osittain yrityksen tiloissa Kempeleessä. Perehdytystä ja tarvittavaa johdatusta iOS-kehitykseen antaa ammattikorkeakoulu.
Kyseinen työ kiinnostaa, koska meillä ei ole ollut koulun kursseilla juurikaan opastusta Applen laitteisiin ja tämä olisi uusi haaste oppia uudesta käyttöjärjestelmästä, uudesta kehitysympäristöstä sekä kohdelaitteesta. Lisäksi aihe sivuaa hiukan sulautettua puolta josta minulla on aikaisempaa työkokemusta.

Tavoite

Työn tavoitteena on perehtyä iOS-käyttöliittymän kehittämiseen ja Fonel-60 järjestelmän käyttämään tekstiviesti-pohjaiseen kommunikaatioon. Työssä hyödynnetään aikaisempaa Android-sovellusta, joka toimii pohjana valmistettavalle iPhone-sovellukselle. Yksi keskeisimmistä ongelmista on siirtää tekstiviestein tapahtuva kommunikointi iOS-ympäristöön, joka ei salli tekstiviestien lähetystä tai vastaanottoa applikaatioista.

Lopputuotoksena pitäisi olla toimiva iOS-applikaatio, tai ainakin käyttöliittymä ohjelmalle jos laitekommunikaatioon liittyvään ongelmaan ei löydetä ratkaisua. Tavoitteena itselleni on oppia kehityksestä Applen laitteilla, iOS-mobiilikehityksestä sekä älylaitteen ja sulautetun laitteen välisestä kommunikaatioista.

Tiedonhankinta

Lähdeaineistoa iOS-kehittämiseen löytyy paljon eripuolilta internettiä. Joitain mainitsemisen arvoisia lähteitä työn ja kirjallisen osan toteuttamiseen:

- FSM-Group. "Fonel-60 asennus ja käyttöohje, versio 1.0."

- Laitinen, Kari 2014. "Harjoituksia iOS-ohjelmointiin liittyen." Oulu, OAMK.
  - OAMK:n iOS-kurssin materiaalia.

- "iOS Dev Center – Apple Developer", 2015. Viitattu 22.02.2015. https://developer.apple.com/devcenter/ios/index.action
  - Ohjeita, frameworkkien liitteet, tutoriaaleja jne.

- "Free iOS and iPhone Programming Course for Beginners | Appcoda", 2015. Viitattu 22.02.2015. http://www.appcoda.com/ios-programming-course/
  - Laaja, ilmainen ja moniosainen tietopaketti iOS-kehitykseen.

Menetelmät

Kehittäminen tapahtuu Applen Xcode ohjelmointiympäristöllä, jolla voi tehdä muun muassa OS X - sekä iOS -ohjelmia. Ohjelmointikielenä tullaan käyttämään joko Objective-C:tä tai uutta Swift-kieltä. Ohjelman käyttöliittymäpuoli tullaan tekemään Xcoden omalla "Interface Builder"

Oulun Ammattikorkeakoulu
Tekniikan yksikkö, Kotkantie
Joni Tauriainen

20.02.2015

-työkalulla käyttäen iOS:n natiivi käyttöliittymäobjekteja.
Projektihallintaan käytetään kevyttä Scrum -viitekehystä taikka Scrum-bania. Scrumissa projektin tekijät pilkotaan resursseiksi ja tuotteen halutuista ominaisuuksista kootaan ominaisuuslista, jossa ominaisuuksille annetaan prioriteetit sekä arvio toteutuksen kestosta. Koko projektille varattu aika pilkotaan sprinteiksi, joihin kootaan tehtäviä halutuista ominaisuuksista. Sprinteistä pidetään suunnittelu-, aloitus- sekä lopetuspalaverit. Scrum-ban on Scrumista kevyempi projektihallinnan viitekehys, jossa ei ole itsenäisiä sprinttejä ja ominaisuuslistaa sekä ominaisuuksien prioriteetteja päivitetään ja muutetaan tarpeen mukaan. Palavereja pidettäisiin säännöllisesti ja tarpeen vaatiessa.

Resurssit

Oumanilla 16.02. käydyn keskustelun perusteella työn suoritukseen tarvittavat laitteet, ohjelmistot ja lisenssit:
- Apple Mac -tietokone, jossa seuraavat ohjelmat
  - Xcoden uusin versio
  - iOS SDK
- "iOS Developer Program" -lisenssi, $99/vuosi
- iPhone, testilaite
- Toimiva Fonel-60 kaappi testausta varten
- Android-puhelin, jolla tutkia Android-versiota ohjelmasta (löytyy jo tekijältä)

Aikataulu

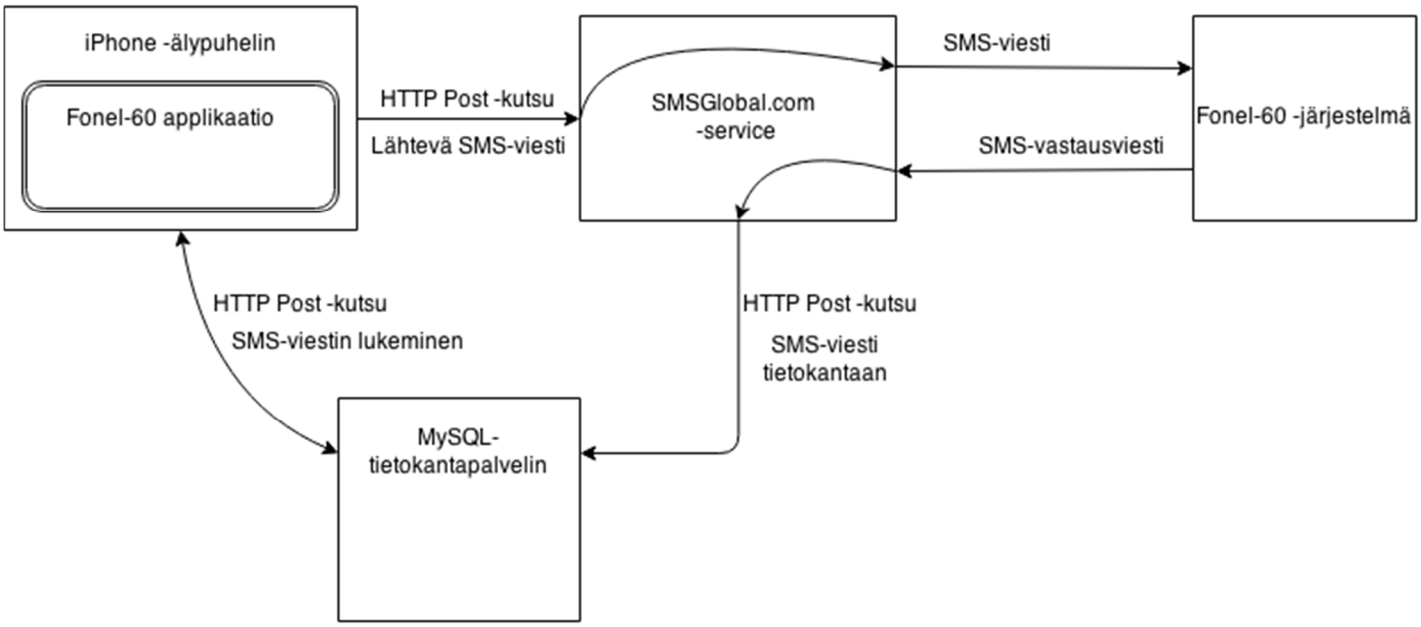
Alustava suunnitelma aikataululle, lopullinen ja tarkempi aikataulu sovitaan aloituspalaverissa.
Projektin aloitus ja suunnitteluvaihe 23.02. - 01.03. (Aloituspalaveri)
Työn aloitus, perehtyminen ja projektisuunnitelma 02.03. - 08.03.
Työn toteutus 09.03. - 18.05. (Raportin palautus)

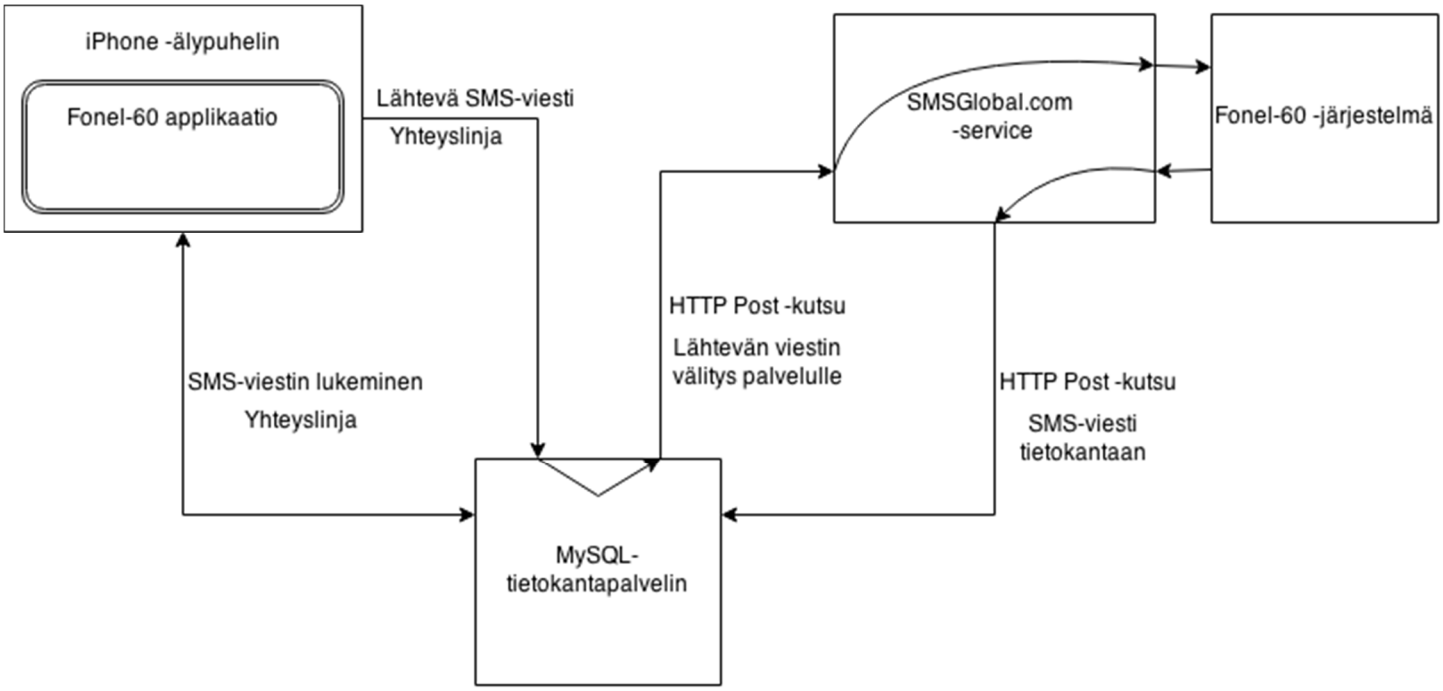
Yhteyshenkilöt

Työn tekijä
Joni Tauriainen
OAMK TTE1SNO
[redacted]

Ohjaava opettaja
Pertti Heikkilä
OAMK
[redacted]

Työn valvoja
Johannes Nikula
Ouman Oy
[redacted]


iPhone -älypuhelin
Fonel-60 applikaatio
HTTP Post -kutsu
Lähtevä SMS-viesti
SMSGlobal.com
-service
SMS-viesti
SMS-vastausviesti
Fonel-60 -järjestelmä
HTTP Post -kutsu
SMS-viestin lukeminen
HTTP Post -kutsu
SMS-viesti
tietokantaan
MySQL-
tietokantapalvelin


iPhone -älypuhelin
Fonel-60 applikaatio
Lähtevä SMS-viesti
Yhteyslinja
SMSGlobal.com
-service
Fonel-60 -järjestelmä
HTTP Post -kutsu
Lähtevän viestin
välitys palvelulle
SMS-viestin lukeminen
Yhteyslinja
HTTP Post -kutsu
SMS-viesti
tietokantaan
MySQL-
tietokantapalvelin